

Esonero Fondamenti Programmazione Canale I 1/12/2010
Soluzioni

Problema 1

Sia T un vettore di N interi e b un intero $b \leq N$. T si dice **b-periodico** se gli elementi del vettore che si trovano in posizioni i aventi lo stesso resto nella divisione intera per b sono uguali. Per esempio è facile verificare che $T=[4,1,2,4,1,2,4,1]$ è 3-periodico.

Progettare una funzione iterativa che dati in input T ed N , la sua dimensione, e $b \leq N$, dica se T è b -periodico.

Il progetto deve descrivere

- idea della soluzione (al massimo tre righe);
- Pre- e post-condizione della funzione;
- variabili usate e loro significato;
- loro inizializzazione;
- proseguimento (o corpo del ciclo) ;
- condizione di terminazione del ciclo;
- verifica della terminazione del ciclo;
- si scriva infine il programma in C che implementa la soluzione.

Idea:

Controllo gli elementi del vettore a partire da quello in posizione b mentre non termino il vettore oppure non trovo che un elemento in posizione i è differente da quello in posizione $T[i \text{ MOD } b]$.

Pre e Post condizione

int f(int T[], int N, int b)

Pre: $N > 0, b \leq N$

Post: $f(T, N, b) = 1$ sse T è b -periodico

Variabili

un intero i , usato come indice su T

un intero flag, usato per controllare se T è b -periodico

Inizializzazione

$i = b$, perchè non ho necessità di vedere i primi elementi $b-1$ elementi

flag = 1, assumo che il vettore si b -periodico e quindi se non ho controllato elementi risulta vero

Proseguimento Controllo se l'elemento in posizione i è uguale all'elemento in posizione $i \text{ MOD } b$. Se non lo è allora il vettore non è b -periodico e quindi flag = 0. Per proseguire con il seguente elemento avanzo i di 1

Condizione di Terminazione. Termino quando ho analizzato tutti gli elementi, quindi $i = N$, oppure quando flag = 0.

Verifica Terminazione. Se T è b -periodico, l'incremento di i ad ogni passo mi garantisce che arrivo a $i = N$. Altrimenti flag sarà falso quando incontro un elemento che non verifica la b -periodicità.

Programma

```
int i=b, flag=1
while(i<N && flag){
    flag = (T[i] == T[i %b]);
    i++;
}
return(flag)
```

oppure (senza flag)

```
int i=b, flag=1
while(i<N && (T[i] == T[i %b]) ) i++;
return(i>=N);
```

Problema 2 (14/30)

Un numero naturale si dice **crescente** se leggendo le sue cifre da sinistra verso destra, si produce una sequenza crescente. Ad esempio $n = 113$ è crescente, mentre $m = 4241$ non lo è.

Progettare una funzione **iterativa** `int crescente(int n, int m)`, che dati due naturali n ed m crescenti e differenti da 0, calcoli un nuovo intero t ottenuto concatenando le cifre di n ed m in maniera crescente. Ad esempio se $n = 133$ ed $m = 234$, t deve essere 123334.

Il progetto deve descrivere

- idea della soluzione (al massimo tre righe);
- Pre- e post-condizione della funzione;
- variabili usate e loro significato;
- loro inizializzazione;
- proseguimento (o corpo del ciclo);
- condizione di terminazione del ciclo;
- verifica della terminazione del ciclo;
- si scriva infine il programma in C che implementa la soluzione.

Soluzione

Idea:

Costruisco il numero risultato a partire dalla cifra meno significativa. Siccome i numeri sono crescenti, questa sarà la maggiore che appare nei due numeri in input. Ripeto questo procedimento, aggiornando il numero in input ed eliminando la cifra trattata. Per poter costruire il numero in output tengo traccia del numero di cifre da cui è composto. Una volta terminato il trattamento di uno dei due numeri in input proseguo con tutto ciò che rimane dell'altro

Variabili

`int ris`, numero in output
`int cent`, contiene il valore $10^{\{\text{\#cifre contate}\}}$

Inizializzazione

$n=0$, non ho analizzato neanche un numero
 $cent = 1$, ho analizzato 0 cifre e quindi $10^0 = 1$

Proseguimento

controllo la maggiore tra le cifre meno significative di n ed m ($n \text{ MOD } 10$ e $m \text{ MOD } 10$), supponiamo $n \text{ MOD } 10$. Aggiorno `ris` aggiungendogli questa cifra, e considerando `cent` ($ris = cent * n \text{ MOD } 10 + ris$). Aggiorno n ($n = n/10$) e `cent` ($cent = 10 * cent$). Nel caso $n \text{ MOD } 10$ e $m \text{ MOD } 10$ siano uguali, ripeto le istruzioni nei due casi.

Condizione Terminazione

Termino (il primo ciclo) non appena uno dei due numeri in input arriva ad essere 0 ($n == 0$ oppure $m == 0$). Eseguo quindi immediatamente un altro ciclo per terminare l'altro numero.

Verifica Terminazione

Certa in quanto ad ogni passo divide almeno uno dei due numeri per 10.

Programma

```
int cent = 1;
int ris = 0;

while (n != 0 && m != 0){
    if (n % 10 > m % 10){
        ris = cent*(n%10)+ris;
        n /= 10;
        cent *= 10;
    } else if (n % 10 < m % 10){
```

```

        ris = cent*(m%10)+ris;
        m /=10;
        cent *=10;
    } else{
        ris = cent*(n%10)+ris;
        n /=10;
        cent *=10;
        ris = cent*(m%10)+ris;
        m /=10;
        cent *=10;
    }

    while (n !=0){
        ris = cent*(n%10)+ris;
        n /=10;
        cent *=10;
    }
    while (m !=0){
        ris = cent*(m%10)+ris;
        m /=10;
        cent *=10;
    }

    return ris;

```

Problema 3 (5/30)

Descrivere la post-condizione della seguente funzione, sapendo che N è un intero ≥ 0 .

```

int f(int N){
    int i,R = 1;
    for(i=0;i<N;i++) R *= 2;
    return R;
}

```

Soluzione

Post: $f(N)=2^N$.

Spazio per Soluzioni